A model for divergence insensitive properties of lambdaY-terms

Sylvain Salvati, Igor Walukiewicz

A model for divergence insensitive properties of λY-terms.

S. Salvati and I. Walukiewicz

Université de Bordeaux, INRIA, CNRS, LaBRI UMR5800
LaBRI Bât A30, 351 crs Libération, 33405 Talence, France

Abstract

A term of a simply typed λ-calculus with fixpoints can be considered as an abstraction of a higher-order functional program. The result of the computation of a term is its Böhm tree. Given a tree automaton describing a property of Böhm trees, we are interested in constructing a model recognizing the property, in a sense that the value of a term determines if its Böhm tree satisfies the property.

We show how to construct models recognizing properties expressed by parity automata that cannot detect divergence. We call them Ω-blind parity automata, as the symbol Ω is used in Böhm trees to represent divergence; an automaton is Ω-blind when it has to accept Ω from every state. The models we construct resemble standard Scott models of latices of monotone functions, but application needs to be modified and the the fixpoint operator should be interpreted as a particular non-extremal fixpoint in a lattice.

1 Introduction

Programs are usually interpreted in infinite models, which makes it in general impossible to calculate their value. Sometimes it is possible to construct a finite model for a given property: the value of a program in the model tells us whether the program has the property or not. This is a particularly pleasant situation as it yields, among others, a modular approach to the decision problem thanks to the structure of the model.

In our case programs are λY-terms: simply typed terms constructed with application, abstraction, and the fixpoint operator. The value of a term is its potentially infinite normal form known as Böhm tree of the term. Under some assumptions on the signature, the Böhm tree of a term of type 0 is just a ranked tree. So we can use parity automata on infinite trees to express properties of terms. In a Böhm tree a special symbol Ω is used to mark places where the computation diverges without producing any visible output. A parity automaton is Ω-blind if it has to accept at Ω.

We present a finite model construction for properties expressed by Ω-blind automata. For a given automaton we construct a model that *recognizes* the property: the value of a term in a model determines if the Böhm tree of the term is accepted by the automaton.

This model construction can be used to solve the higher-order model checking problem. The problem has been shown decidable by Ong [Ong06]. The result has then been reproved in a number of different ways. The most relevant for this paper are typing approach of Kobayashi and Ong [KO09], and our approach through Krivine machines [SW14]. Both approaches actually work with Ω-blind automata. If one is interested solely in decidability then this is sufficient since it is possible to transform a term to a "similar" term without divergent computations.

The issue of detecting divergence directly was our first example of the use of model approach. In [SW13] we have considered the case of safety properties together with divergence detection. Later, we have extended this approach to weak tree automata [SW15b]. Finally, recently we have given a model construction for all parity automata [SW15a]. None of these models can be seen as a special case of the model presented here. Indeed, as it is made clear in [SW15a], an additional property that we call stratification is required.

The case of Ω-blind automata studied here is interesting because it has been considered in a number of papers, and because the model construction is simpler than in the general case. For example Tsukada and Ong [TO14] give a type system and an infinite model for Ω-blind automata. Grelois and Meilles derive an infinite model [GM15b, GM15c] from a general linear logic constructions. More recently they have obtained also a finite model [GM15a]. The contribution of this paper is to present an arguably simple model construction for Ω-blind automata. It produces almost standard Scott models where a type is interpreted as a lattice of monotone functions: the application operation is modified, and the correct meaning of fixpoint needs to be found.

Organisation of the paper: The next section introduces λY-calculus, Böhm trees, and parity automata running on Böhm trees. In particular, it defines the Ω-blindness condition, and discusses some of its consequences. The main result is also stated in this section. Section 3 reduces the question of whether an automaton $\mathcal{A}$ accepts a Böhm tree $BT(M)$ to the existence of a winning strategy in a game $\mathcal{K}(\mathcal{A}, M)$ that is defined in that section. Section 4 introduces a finite game $\mathcal{G}(\mathcal{A}, M)$ and shows its equivalence with $\mathcal{K}(\mathcal{A}, M)$. Section 5 proves that Eve can play with monotone functions in $\mathcal{G}(\mathcal{A}, M)$; this gives us a variant of the game that we call $\mathcal{G}^{mon}(\mathcal{A}, M)$. Finally, Section 6 defines a finitary model for a fixed automaton $\mathcal{A}$, and shows a direct correspondence between the value of a term in this model

and winning in a game $\mathcal{G}^{mon}(\mathcal{A}, M)$.

2 Preliminaries

We start by introducing λY-calculus and Böhm trees as the results of evaluation of λY-terms. Then we introduce parity automata on infinite trees. We will use them to accept Böhm trees. We formulate the Ω-blind restriction on transitions of the automata that is central for this paper. Finally, we state the main result of the paper, and give a detailed outline of the construction.

2.1 λY-calculus and Böhm trees

The *set of types* is constructed from a unique *basic type* o using a binary operation $\to$ that associates to the right. Thus o is a type, and if A, B are types, so is $(A \to B)$. We use $Types$ to denote the set of all types. The order of a type is defined by: $order(o) = 0$, and $order(A \to B) = max(1 + order(A), order(B))$. We work with *tree signatures* that are finite sets of *typed constants of order at most* 1. Types of order 1 are of the form $o \to \cdots \to o \to o$ that we abbreviate $o^i \to o$ when they contain $i + 1$ occurrences of o. For convenience we assume that $o^0 \to o$ is just o. If Σ is a signature, we write $\Sigma^{(i)}$ for the set of constants of type $o^i \to o$.

Simply typed λY-terms are built from the constants in the signature, and constants Y^A for every type A. These stand for the *fixpoint combinator* and *undefined term*, respectively. Apart from constants, for each type A there is a countable set of variables $x^A, y^A, \ldots$. Terms are built from these constants and variables using typed application and λ-abstraction. We shall write sequences of λ-abstractions $\lambda x_1 \ldots \lambda x_n.\ M$ with only one λ: either as $\lambda x_1 \ldots x_n.\ M$, or even shorter as $\lambda \vec{x}.\ M$.

The usual operational semantics of the λ-calculus is given by β-contraction. To give the meaning to fixpoint constants we use δ-contraction ($\to_\delta$):

$$(\lambda x.M)N \to_\beta M[N/x] \qquad \text{and} \qquad YM \to_\delta M(YM)\ .$$

Remark: The status of Y constant in the calculus is a bit special: it is a constant, but it is a subject of a reduction rule. In our game formulations to simplify the presentation we will assume that Y always appears in an applicative context, i.e., in a form YN for some N. By replacing Y with $\lambda x.\ Yx$ we can transform any term to a term with this property. We will only consider extensional models in this paper, and in such models the meaning of Y and $\lambda x.Yx$ is the same. So every term can be transformed to an equivalent term where Y is appears only in an applicative context.

The *Böhm tree* of a term M is obtained by first reducing it until obtaining the head normal form, i.e., a term of a shape $\lambda \vec{x}.N_0N_1 \ldots N_k$ with N_0 a variable or a constant; in this case $BT(M)$ is a tree having its root labelled

by $\lambda\vec{x}.N_0$ and having $BT(N_1), \ldots, BT(N_k)$ as subtrees. If M does not have a head normal form then $BT(M) = \Omega^A$, where Ω^A is a special symbol of the same type as M. Thus Böhm trees are infinite normal forms of λY-terms.
Remark: A Böhm tree of a closed term of type o over a tree signature is a potentially infinite ranked tree: a node labelled by a constant a of type $o^i \to o$ has i successors. Among symbols Ω^A, only the symbol Ω^o can appear in the Böhm tree of such a term.

2.2 Parity automata accepting Böhm trees

Automata will work on Σ-labelled trees, where Σ is a tree signature. Trees are partial functions $t : \mathbb{N}^* \dot{\to} \Sigma \cup \{\Omega\}$ such that the number of successors of a node is determined by the label of the node. In particular, if $t(u) \in \Sigma^{(0)}$ then u is a leaf. The *nodes* of t, are the elements of the domain of t. The set of nodes should be prefix closed. A *label* of a node u is $t(u)$.

We will use nondeterministic max-parity automata, that we will call *parity automata* for short. Such an automaton accepts trees over a fixed tree signature Σ. It is a tuple

$$\mathcal{A} = \langle Q, \Sigma, \{\delta_i\}_{i\in\mathbb{N}}, rk : Q \to [m]\rangle$$

where Q is a finite set of states, rk is the *rank function* with the range $[m] = \{0, \ldots, m\}$, and $\delta_i : Q \times \Sigma^{(i)} \to \mathcal{P}(Q^i)$ is the *transition function*. Observe that since the signature Σ is finite, only finitely many δ_i are nontrivial. From the definition it follows that, for example, $\delta_2 : Q \times \Sigma^{(2)} \to \mathcal{P}(Q \times Q)$ and $\delta_0 : Q \times \Sigma^{(0)} \to \{\emptyset, \{\emptyset\}\}$. We will simply write δ without a subscript when this causes no ambiguity.

In order to accept Böhm trees over Σ we need also to define the behaviour of $\mathcal{A}$ on the special symbol Ω^o denoting divergence. In this paper we put an important restriction on the transition function:

Ω-blind: $\delta(\Omega^o, q) = \{\emptyset\}$ for all states q.

As we will see this means that the part of the run ending in Ω is always accepting. We will discuss consequences of this restriction at the end of this subsection.

A *run of $\mathcal{A}$ on t from a state q^0* is a labelling of nodes of t with the states of $\mathcal{A}$ such that: (i) the root is labelled with q^0, (ii) if a node u is labelled q and its k-successors are labelled by $q_1, \ldots q_k$, respectively, then $(q_1, \ldots, q_k) \in \delta_k(q, t(u))$; recall that $t(u)$ is the letter in the node u.

A run is *accepting* when: (i) for every leaf u of t, if q is the state of the run in u then $\delta_0(q, t(u)) = \{\emptyset\}$, and moreover (ii) for every infinite path of t, the labelling of the path given by the run satisfies the *parity condition*. This means that if we look at the ranks of states assigned to the nodes of the path then the maximal rank appearing infinitely often is even. A tree

is *accepted by* $\mathcal{A}$ *from a state* q^0 if there is an accepting run from q^0 on the tree.

Remark: Let recall that the automata model can be extended to alternating parity automata without increasing the expressive power. Here, for simplicity of the presentation, we will work only with nondeterministic automata but our constructions apply also to alternating automata.

Remark: The Ω-blind restriction has some consequences. The restriction forces automaton to unconditionally accept all diverging computations; so Ω-blind automata cannot express all MSOL properties of Böhm trees. The name Ω-blind has been proposed in [SW13] as opposed to *insightful* automata that can reject Ω from some states. Actually, the expressive power of MSOL is captured by insightful automata that accept Ω precisely in states of an even rank. The type system of Kobayashi and Ong [KO09], as well as its refinement by Tsukada and Ong [TO14], work for Ω-blind automata.

If one is interested purely in the decidability of MSOL model checking of Böhm trees of λY-terms then there are several ways of reducing this problem to the Ω-blind case. On the level of terms, one can transform a λY-term into a λY-term that generates the same tree but for Ω replaced by some fresh constant [Had13, SW13]. There is a simpler solution but it requires a more substantial transformation of the automaton. It is possible to transform a term to an equivalent one where every recursion is guarded: YM is transformed to $Y\lambda\vec{x}.\ e(M\vec{x})$ for e a new constant of type $o \to o$ and $\vec{x}$ a sequence of variables making $M\vec{x}$ of type o. From the strong normalisation property of the simply typed λ-calculus it follows that there are no Ω symbols in a Böhm tree of terms obtained by this transformation. Then it remains to transform an automaton to an automaton that "almost ignores" the fresh e symbols, so that acceptance of the transformed Böhm tree by the transformed automaton is equivalent to the acceptance of the original tree by the original automaton.

2.3 Formulation of the problem and overview of the construction

Our aim is to construct a finite model for a given Ω-blind parity automaton. The model should recognize the set of terms whose Böhm trees are accepted by the automaton. This means that the value of a term in the model should determine if the Böhm tree of the term is accepted by the automaton.

Our models will be built on the set of states of the automaton. In particular, the value of a term of type o will be the set of states from which the automaton accepts the Böhm tree of the term. The main challenge would be to find the appropriate meaning for the fixpoint in order to capture the parity acceptance condition.

Theorem 1 *For every Ω-blind parity automaton $\mathcal{A}$, there is a finite model $\langle \{\mathcal{S}_A\}_{A \in Types}, \{[\![a]\!]\}_{a \in \Sigma}, \{[\![Y^A]\!]\}_{A \in Types} \rangle$, with $\{\mathcal{S}_o\} = \mathcal{P}(Q)$, such that for every closed term M of type o: the semantics of M in the model is the set of states from which $\mathcal{A}$ accepts $BT(M)$.*

We will start with defining an infinite game $\mathcal{K}(\mathcal{A}, M)$ characterising acceptance of the Böhm tree of a term M by automaton $\mathcal{A}$; more precisely Eve will win in this game if $BT(M) \in L(\mathcal{A})$. The game $\mathcal{K}(\mathcal{A}, M)$ is based on the evaluation of M with a Krivine machine.

Then we will construct a finite game $\mathcal{G}(\mathcal{A}, M)$ using a concept of residual. We show that this game is equivalent to $\mathcal{K}(\mathcal{A}, M)$ in the sense that the winners in the two games are the same. This is essentially the construction from [SW11]. This reduction is sufficient to show the decidability of the model-checking problem, but not to obtain the model.

The next step is the restriction to monotone residuals. The residuals in $\mathcal{G}(\mathcal{A}, M)$ can be arbitrary functions. In order to take fixpoints we need to restrict to monotone functions. We show that the resulting game, $\mathcal{G}^{mon}(\mathcal{A}, M)$ is equivalent to $\mathcal{G}(\mathcal{A}, M)$.

The final step is to define a model whose elements are monotone residuals. The model will capture precisely the notion of winning in $\mathcal{G}^{mon}(\mathcal{A}, M)$. As it will be clear from the constructions, the notion of winning in $\mathcal{G}^{mon}(\mathcal{A}, M)$ is defined for terms of all types. This allows for a simple proof of the soundness and the completeness of the model using an induction on the size of the term. The only complicated case is to understand the meaning of the fixpoint in the model.

3 Game $\mathcal{K}(\mathcal{A}, M)$

In order to define the game we first need to introduce the Krivine machine that is a call-by-name evaluation mechanism for λY-terms. Once we explain how the Krivine machine is used to compute Böhm tree, the definition of $\mathcal{K}(\mathcal{A}, M)$ will be obtained as a straightforward reformulation of the acceptance by a tree automaton in terms of games. The only new element will be some bookkeeping for closures that we will use later in the reduction to a finite game. The main result, stated in Proposition 4, says that Eve wins in $\mathcal{K}(\mathcal{A}, M)$ if and only if $BT(M)$ is accepted by $\mathcal{A}$.

In this section we assume that in M the constant Y appears only in applicative contexts, as discussed in the Remark on page 3.

A *Krivine machine* [Kri07], is an abstract machine computing the weak head normal form of a λ-term, using explicit substitutions, called *environments*. Environments are functions assigning *closures* to variables, and closures themselves are pairs consisting of a term and an environment. This

mutually recursive definition is schematically represented by the grammar:

$$C ::= (M, \rho) \qquad \rho ::= \emptyset \mid \rho[x \mapsto C]$$

As in this grammar, we will use $\emptyset$ for the empty environment. We require that in a closure (M, ρ), the environment is defined for every free variable of M. Intuitively such a closure denotes closed λ-term: it is obtained by substituting for every free variable x of M the λ-term denoted by the closure $\rho(x)$.

A configuration of the Krivine machine is a triple (M, ρ, S), where M is a term, ρ is an *environment*, and S is a *stack* (a sequence of closures with the topmost element on the left). The rules of the Krivine machine are as follows:

$$\begin{aligned}
(\lambda x.M, \rho, (N, \rho')S) &\to (M, \rho[x \mapsto (N, \rho')], S) \\
(YM, \rho, S) &\to (M(YM), \rho, S) \\
(MN, \rho, S) &\to (M, \rho, (N, \rho)S) \\
(x, \rho, S) &\to (M, \rho', S) \qquad \text{where } (M, \rho') = \rho(x)
\end{aligned}$$

Note that the machine is deterministic provided that for terms of the form YM the fixpoint rule rather than the application rule is used.

Since we use the Krivine machine on typed terms we require that the environment ρ associates to a variable x of type A a closure (N, ρ') with a term N of type A; we will say that the closure is of type A too. In a configuration (M, ρ, S), if M has type $A_1 \to \cdots \to A_n \to 0$, then $S = C_1 \dots C_n$ is a stack of n closures, with C_i of type A_i.

To compute the Böhm tree of a closed term M of type o we start the Krivine machine in the configuration $(M, \emptyset, \varepsilon)$. The sequence of reductions from this configuration is either infinite or terminates in a configuration of a form $(b, \rho, C_1 \dots C_n)$, where b is a constant of type $o^n \to o$ (this is because we have assumed that the types of constants have rank at most 1). In the former case, the Böhm tree is Ω^o. In the latter case, we create a node labelled b and start reducing in parallel $(N_i, \rho_i, \varepsilon)$ for $i = 1, \dots, k$; where $(N_i, \rho_i) = C_i$. This process gives at the end a tree labelled with constants that is precisely $BT(M)$; that is the object of our study. Notice that if (N, ρ, S) is reachable from $(M, \emptyset, \varepsilon)$ then N, and the terms that occur in ρ and in S are all subterms of M. One should be careful with a definition of a subterm though. Since we have a fixpoint operator we consider that $N(YN)$ is a subterm of YN. Of course even with this twist, the number of subterms of a term remains finite.

The game $\mathcal{K}(\mathcal{A}, M)$ will be based on the reductions of the Krivine machine. The positions of the game will be of the form $q : (N, \rho, S)$ where q is a state of automaton $\mathcal{A}$, and (N, ρ, S) is a configuration of the Krivine machine with a small modification of the notion of closure. Now, a closure

will have one more component; it will have a form (u, N, ρ) where the new component u is a node of the tree being constructed. Basically, the role of u is to identify the closure uniquely.

Definition 2 (The run tree $RT(\mathcal{A}, M)$) This is a tree with the root labelled $q_0 : (M, \emptyset, \varepsilon)$, and constructed according to the following rules. In rule 2 below we put ranks an edges of the tree. These ranks will be used later in the definition of a game.

1. $q : (a, \rho, C_1 \dots C_k) \quad \longrightarrow \quad (q_1, \dots, q_k) : (a, \rho, C_1 \dots C_k)$,
   for every $(q, a, q_1, \dots, q_k) \in \delta_{\mathcal{A}}$;
2. $(q_1, \dots, q_k) : (a, \rho, C_1 \dots C_k) \quad \stackrel{rk(q_i)}{\longrightarrow} \quad (q_i, u_i) : (N_i, \rho_i, \varepsilon)$,
   where $C_i = (u_i, N_i, \rho_i)$ for $i = 1, \dots, k$;
3. $q : (\lambda x.N, \rho, C\vec{C}) \quad \longrightarrow \quad q : (N, \rho[C/x], \vec{C})$;
4. $q : (YN, \rho, \vec{C}) \quad \longrightarrow \quad q : (N(YN), \rho, \vec{C})$;
5. $q : (NK, \rho, \vec{C}) \quad \longrightarrow \quad q : (N, \rho, (u, K, \rho)\vec{C})$;
   where u is the node of the tree where the rule is applied.
6. $q : (x, \rho, \vec{C}) \quad \longrightarrow \quad (q, u) : (N, \rho', \vec{C})$;
   where $\rho(x) = (u, N, \rho')$;
7. $(q, u) : (N, \rho, \vec{C}) \quad \longrightarrow q : (N, \rho, \vec{C})$

We can now define a game on the run tree that will characterise the acceptance of $BT(M)$ by $\mathcal{A}$. The two players, Eve and Adam, will choose a path in the run tree and the winner will be determined by a parity condition. Observe that only the positions from the first two rules may have more than one successor.

Definition 3 (Game $\mathcal{K}(\mathcal{A}, M)$) The game $\mathcal{K}(\mathcal{A}, M)$ is played on $RT(\mathcal{A}, M)$. Eve chooses a transition given by the first rule, Adam a transition given by the second rule. The ranks are defined on edges. If an edge has a label then it is its rank, the edges with no labels have rank 0. A finite play that cannot be prolonged is winning for Eve if its last node is of the form $\emptyset : (c, \rho, \varepsilon)$, otherwise it is winning for Adam. Eve wins an infinite play if the sequence of ranks on the path satisfies the parity condition.

The condition for finite plays in the above definition says that Eve looses if she cannot find a suitable transition of the automaton.

Proposition 4 For every automaton $\mathcal{A}$, and closed term M of type o:

$\mathcal{A}$ accepts $BT(M)$ from q iff Eve wins in $\mathcal{K}(\mathcal{A}, M)$ from $q : (M, \emptyset, \varepsilon)$.

4 Game $\mathcal{G}(\mathcal{A}, M)$

We construct a finite game $\mathcal{G}(\mathcal{A}, M)$ equivalent to $\mathcal{K}(\mathcal{A}, M)$. The main result is that the two games are equivalent, Theorem 9. For this we define a notion of residual for a node in a strategy in $\mathcal{K}(\mathcal{A}, M)$ and show how it can be used to play in $\mathcal{G}(\mathcal{A}, M)$.

In this section we fix a parity automaton $\mathcal{A}$ with the set of states Q, the rank function $rk : Q \to [m]$, and the transition function δ. We also fix a term M and suppose that Y appears only in applicative contexts in M, as explained in the Remark on page 3.

Definition 5 (Residuals) For every type A we define the set of residuals $\mathcal{R}_A$ of type A:

$$\mathcal{R}_o = \mathcal{P}(Q \times [m]), \qquad \mathcal{R}_{A\to B} = R_A \to R_B \ .$$

Definition 6 (Lifting operation) For a residual $R \in \mathcal{R}_o$ and $r \in [m]$ we define

$$R\lfloor_r = \{(q_1, r_1) \in R : r_1 > r\} \cup \{(q_1, r_2) : \exists_{(q_1,r_1)\in R}\ r_2 \leq r_1 = r\}$$

For $R \in \mathcal{R}_{A\to B}$ we define $R\lfloor_r(S) = (R(S))\lfloor_r$.

Lemma 7 For every residual R, and every $r, r_1, r_2 \in [m]$:

- $(R\lfloor_{r_1})\lfloor_{r_2} = R\lfloor_{\max(r_1,r_2)}$.
- $(q, 0) \in R\lfloor_r$ iff $(q, r) \in R$.

Proof
We prove only the first statement. The second statement follows directly from the definitions.

Suppose $r_1 \geq r_2$. We show $(R\lfloor_{r_1})\lfloor_{r_2} = R\lfloor_{r_1}$. Fix a state q and look at all the pairs with this state. The pairs (q, r) with $r > r_1$ are the same in R and $(R\lfloor_{r_1})\lfloor_{r_2}$. If $(q, r_1) \notin R$ then there is no pair (q, r) in $R\lfloor_{r_1}$ with $r \leq r_1$, and $\lfloor_{r_2}$ does nothing for pairs with state q. If $(q, r_1) \in R$ then all the pairs (q, r) for $r \leq r_1$ are in $R\lfloor_{r_1}$. The operation $\lfloor_{r_2}$ does not add or remove any pairs with the state q.

When $r_1 < r_2$ we show that $(R\lfloor_{r_1})\lfloor_{r_2} = R\lfloor_{r_2}$. The proof is similar. □

Definition 8 (Game $\mathcal{G}(\mathcal{A}, M)$) The positions of the game are of the form $(N, \theta) \geq \vec{R} \to q$ where N is a subterm of M, θ is a function mapping variables of M to residuals of appropriate types, $\vec{R}$ is a sequence of residuals of types determined by the type of N, and q is a state of $\mathcal{A}$. The game starts in the position $(M, \emptyset) \geq q$, and proceeds according to the following rules:

$$\frac{(N,\theta[P/x]) \geq \vec{R} \to q}{(N,\theta) \geq \lambda x.N : P \to \vec{R} \to q} \qquad \frac{(N(YN),\theta) \geq \vec{R} \to q}{(YN,\theta) \geq \vec{R} \to q}$$

$$\frac{(N,\theta) \geq P \to \vec{R} \to q \qquad (K,\theta) \geq P}{(NK,\theta) \geq \vec{R} \to q}$$

$$\frac{(K,\theta) \geq_r \vec{R} \to q \quad \text{for all } \vec{R} \text{ and } (q,r) \in P(\vec{R})}{(K,\theta) \geq P} \qquad \frac{(K,\theta\downarrow_r) \geq \vec{R} \to q}{(K,\theta) \geq_r \vec{R} \to q}$$

$$\frac{\text{if } (q,0) \in \theta(x)(\vec{R})}{(x,\theta) \geq \vec{R} \to q}$$

$$\frac{\text{if there is } (q_1 \ldots, q_k) \in \delta_{\mathcal{A}}(q,a) \text{ so that } (q_i,0) \in R_i\downarrow_{rk(q_i)} \text{ for all } i = 1,\ldots k}{(a,\theta) \geq R_1 \to \cdots \to R_k \to q}$$

The shape of the position determines the rule to be applied. Moreover, the rules for λ-abstraction and Y fixpoint are deterministic. In the case of the application rule, Eve chooses a residual P and Adam chooses one of the premisses. In the case of the residual rule, Adam chooses one of the premisses. Positions with rank subscripts of the form $(K,\theta) \geq_r \vec{R} \to q$ are only used to define ranks of positions; see below. The corresponding rule tells that the rank acts on the environment. The last two rules are termination conditions: for the case of a variable, and a constant other that Y, respectively. Recall that we assume that Y appears always in the applicative context (cf. Remark on page 3) so we do not need a rule for Y appearing separately.

Ranks are defined by:

- positions of the form $(K,\theta) \geq_r \vec{R} \to q$ have rank r;
- other positions have rank 0.

The game $\mathcal{G}(\mathcal{A},M)$ is clearly finite. Next theorem states that it is equivalent to the infinite game $\mathcal{K}(\mathcal{A},M)$.

Theorem 9 *For every closed term M of type o, every automaton $\mathcal{A}$ and its state q:*

Eve wins from $q : (M,\emptyset,\varepsilon)$ in $\mathcal{K}(\mathcal{A},M)$ iff Eve wins from $(M,\emptyset) \geq q$ in $\mathcal{G}(\mathcal{A},M)$

The rest of this section is devoted to the proof of the theorem. We start with an important definition of a residual associated to a node of the tree.

We consider subtree $\mathcal{T}$ of $\mathcal{K}(\mathcal{A},M)$, and define residuals $R_{\mathcal{T}}(u)$ and $res_{\mathcal{T}}(u,u')$ for some nodes u,u' in $\mathcal{T}$. We will use this definition in cases when $\mathcal{T}$ represents a strategy for Eve or for Adam in $\mathcal{K}(\mathcal{A},M)$. We will

omit $\mathcal{T}$ subscript, as $\mathcal{T}$ will be clear from the context. The residual $R(u)$ will be defined for all application nodes in $\mathcal{T}$, that is nodes u with labels of the form $(NK, \theta) \geq \vec{R} \to q$.

For a node u and its descendant u' we write $\max(u, u')$ for the maximum of the ranks of edges on the path from u to u'.

Definition 10 (Residuals of a node: $R(u)$) For $\mathcal{T}$ a subtree of $\mathcal{K}(\mathcal{A}, M)$ we define a residual $R_{\mathcal{T}}(u)$ for every application node u of $\mathcal{T}$. The definition is by induction on the type of the closure created by the application rule.

Consider a closure (u, K, ρ) with K of type o. The residual $R_{\mathcal{T}}(u) \in \mathcal{R}_o$ is the set of pairs $(q, \max(u, u'))$ such that there is in $\mathcal{T}$ a node u' labelled with $(q, u) : (K, \rho, \emptyset)$.

Before giving the definition for other types we will introduce some abbreviations. For u' a descendant of u, we define $res(u, u')$ to be $R(u)\lfloor_{\max(u,u')}$. Similarly, for a closure (u, K, ρ) we define $res((u, K, \rho), u') = res(u, u')$. We then extend this operation to sequences of closures: $res(\vec{C}, u')$ is the sequence of residuals obatined by applying $res(\cdot, u')$ componentwise to every element of $\vec{C}$.

Now, consider a closure (u, K, ρ) of an arbitrary type $A \equiv A_1 \to \cdots \to A_k \to o$. The residual $R_{\mathcal{T}}(u) \in \mathcal{R}_A$ is a function such that for every $\vec{S} \in \mathcal{R}_{\mathcal{A}_1} \times \cdots \times \mathcal{R}_{\mathcal{A}_k}$ the set $R_{\mathcal{T}}(u)(\vec{S})$ contains a pair $(q, \max(u, u'))$ if there is a node u' in $\mathcal{T}$ labelled by $(q, u) : (K, \rho, \vec{C})$ with $res(\vec{C}, u') = \vec{S}$.

We will now show how to transfer Eve's winning strategy in $\mathcal{K}(\mathcal{A}, M)$ to her winning strategy in $\mathcal{G}(\mathcal{A}, M)$. Then we will do the same for strategies of Adam.

4.1 Transferring Eve's strategy from $\mathcal{K}(\mathcal{A}, M)$ to $\mathcal{G}(\mathcal{A}, M)$

We will use a winning strategy σ_{Eve} for Eve in $\mathcal{K}(\mathcal{A}, M)$ to show how Eve can win in $\mathcal{G}(\mathcal{A}, M)$. When in a position v in $\mathcal{G}(\mathcal{A}, M)$ we will keep an associated position u in $\mathcal{K}(\mathcal{A}, M)$. Then looking at u we will find a successor v' of v and associate to it a node u' of $\mathcal{K}(\mathcal{A}, M)$ that is reachable when playing σ_{Eve} from u. The pairs of nodes will satisfy the following invariant:

> u is labelled by $q : (N, \rho, \vec{C})$, and v is labelled by $(N, \theta) \geq \vec{R} \to q$ with $\theta = res(\rho, u)$, and $\vec{R} = res(\vec{C}, u)$.

Above, and in the rest of this subsection, all residuals $R(u)$, $res(u, u')$, are calculated with respect to σ_{Eve}, i.e., we take σ_{Eve} as $\mathcal{T}$ in definitions above.

We will now show how Eve can play in order to preserve the above invariant. For this we will examine the rules one, by one depending on the shape of the term N.

The case of a variable Suppose the play reaches a node v with a variable in the label. The invariant says that we have a companion node u with the following situation:

$$u \text{ is } q : (x, \rho, \vec{C}) \qquad v \text{ is } (x, \theta) : \vec{R} \to q$$
$$\theta = res(\rho, u) \quad \text{and} \quad \vec{R} = res(\vec{C}, u) \ .$$

We need to show that v is a winning position for Eve, namely that $(q, 0) \in \theta(x)(\vec{R})$. The value $\rho(x)$ is some closure of the form (u_x, N, ρ'). So the successor u' of the position u in σ_{Eve} is labelled by $(q, u_x) : (N, \rho', \vec{C})$. Using the definition of $R(u_x)$ we obtain $(q, \max(u_x, u')) \in R(u_x)(\vec{R})$, as induction assumption gives us $\vec{R} = res(\vec{C}, u)$, and $res(\vec{C}, u) = res(\vec{C}, u')$ since the rank of the transition from u to u' is 0. By Lemma 7, $(q, 0) \in (R(u_x)(\vec{R}))\lfloor_{\max(u_x, u')}$. Since $\max(u_x, u') = \max(u_x, u)$ we obtain

$$(q, 0) \in (R(u_x)(\vec{R}))\lfloor_{\max(u_x,u)} = (R(u_x)\lfloor_{\max(u_x,u)})(\vec{R}) = \theta(x)(\vec{S}) \ .$$

The case of a constant Suppose the play reaches a node v with a constant in the label. The invariant tells us that we have an associated node u with the following situation:

$$u \text{ is } q : (a, \rho, C_1 \dots C_k) \qquad v \text{ is } (a, \theta) : R_1 \to \dots \to R_k \to q$$
$$\theta = res(\rho, u) \quad \text{and} \quad R_i = res(C_i, u) \text{ for } i = 1, \dots k$$

We show that v is a winning position for Eve. For this we need to find $(q_1, \dots, q_k) \in \delta_{\mathcal{A}}(q, a)$ such that $(q_i, rk(q_i)) \in R_i\lfloor_{rk(q_i)}$ for all $i = 1, \dots, k$.

Since u is a part of the winning strategy σ_{Eve} for Eve, it has a successor u' in σ_{Eve} labelled $(q_1, \dots, q_k) : (a, \rho, C_1 \dots C_k)$ where $(q_1, \dots, q_k) \in \delta_{\mathcal{A}}(q, a)$. This is the transition we take. If $k = 0$ this tells us that $\emptyset \in \delta_{\mathcal{A}}(q, a)$ so Eve wins. In the remaining we will suppose $k > 0$.

Let us see what we can deduce about R_i. By the invariant $R_i = res(C_i, u)$. Now C_i is a closure of the form (u_i, N_i, ρ_i), and by definition $res(C_i, u) = R(u_i)\lfloor_{\max(u_i,u)}$. So we need to examine $R(u_i)$.

For every $i = 1, \dots, k$, position u' has a successor u'_i labelled with $(q_i, u_i) : (N_i, \rho_i, \varepsilon)$. The transition from u' to u'_i has rank $rk(q_i)$. By definition of $R(u_i)$, we have that the node u'_i contributes $(q_i, \max(u_i, u'_i))$ to $R(u_i)$. Observe that $\max(u_i, u'_i) = \max(rk(q_i), \max(u_i, u))$.

From these observations we get $(q_i, \max(rk(q_i), \max(u_i, u'_i))) \in R(u_i)$ and $R_i = R(u_i)\lfloor_{\max(u_i,u)}$. Then $R_i\lfloor_{rk(q_i)}$ is $R(u_i)\lfloor_{\max(rk(q_i),\max(u_i,u)}$ by Lemma 7. So directly from the definition of $\lfloor_r$ operation $(q_i, 0) \in R_i\lfloor_{rk(q_i)}$. This shows that v is a winning position for Eve.

The case of application Suppose that v is an application node. The invariant gives us a companion node u so that:

$$u \text{ is } q : (NK, \rho, \vec{C}) \qquad v \text{ is } (NK, \theta) : \vec{R} \to q$$
$$\theta = res(\rho, u) \quad \text{and} \quad \vec{R} = res(\vec{C}, u)$$

Node u has the unique successor u' labelled by $q : (NK, \rho, (u, K, \rho)\vec{C})$. In order to proceed from the node v, Eve should use the application rule with $P = R(u)$. After this Adam can choose between two positions:

$$(NK, \theta) \geq R(u) \to \vec{R} \to q \quad \text{and} \quad (K, \theta) \geq R(u) \ .$$

If Adam chooses the first one, call it v', then we can take u' as a companion node satisfying the invariant. Indeed, since the rank of the transition from u to u' is 0 we get: $res(R(u), u') = R(u)\lfloor_0 = R(u)$, $\theta = res(\rho, u')$, and $\vec{R} = res(\vec{C}, u')$.

If Adam chooses $(K, \theta) \geq R(u)$ then in the next step he can choose some $\vec{S}$ and some $(q', r') \in R(u)(\vec{S})$. In this case the new position in the game is $(K, \theta) \geq_{r'} \vec{S} \to q'$, and the position after it is $(K, \theta\lfloor_{r'}) \geq \vec{S} \to q'$. Call this position v_2.

We need to find a descendant u_2 of u that we can take as a companion for v_2. Since $(q', r') \in R(u)(\vec{S})$, there is a descendant u_1 of u labelled by $(q, u) : (K, \rho, \vec{C_1})$ and moreover $\vec{S} = res(\vec{C}, u_1)$, and r' is the maximal rank on the path from u to u_1. Node u_1 has the unique successor labelled by $q : (K, \rho, \vec{C_1})$. This is the node u_2 that we take as a companion of v_2. To see that the invariant is verified we note that the rank of the transition form u_1 to u_2 is 0 giving us: $\theta\lfloor_{r'} = res(\rho, u)\lfloor_{r'} = res(\rho, u_2)$, and $\vec{S} = res(\vec{C_1}, u_1) = res(\vec{C_1}, u_2)$.

Abstraction and fixpoint These rules are deterministic and it is direct to verify that the invariant is preserved.

Every infinite play is winning The cases of a variable and a constant treated above covered all possibilities of terminating plays. If a play following the strategy described above is infinite then we obtain a sequence of pairs of positions $(u_1, v_1)(u_2, v_2) \ldots$, such that $v_1, v_2, \ldots$ is the play in $\mathcal{G}(\mathcal{A}, M)$, and $u_1, u_2, \ldots$ is a sequence of nodes on a path of σ_{Eve}. We also obtain that the rank of a node v_i is the same as the maximal rank on the path from u_{i-1} to u_i. So since the σ_{Eve} is a winning strategy, the maximal ranks appearing on the play $v_1, v_2, \ldots$ is even.

4.2 Transferring Adam's strategy from $\mathcal{K}(\mathcal{A}, M)$ to $\mathcal{G}(\mathcal{A}, M)$

We will proceed in the same way with strategies of Adam. We take a winning strategy σ_{Adam} of Adam in $\mathcal{K}(\mathcal{A}, M)$. To every position v in $\mathcal{G}(\mathcal{A}, M)$

reached during a play we will associate a position u in $\mathcal{K}(\mathcal{A}, M)$. The two positions will satisfy a similar invariant, but to formulate it we need one more definition.

Definition 11 (Complementarity predicate) For two residuals $R_1, R_2 \in \mathcal{R}_A$ of the same type we define when they are complementary by induction on the rank of the type:

- for $A = o$, we put $\mathit{Comp}(R_1, R_2)$ if $R_1 \cap R_2 = \emptyset$,
- for $A = B \to C$, we put $\mathit{Comp}(R_1, R_2)$ if for all $S_1, S_2 \in \mathcal{R}_B$ such that $\mathit{Comp}(S_1, S_2)$ we have $\mathit{Comp}(R_1(S_1), R_2(S_2))$.

Lemma 12 If $\mathit{Comp}(R_1, R_2)$ then $\mathit{Comp}(R_1\restriction_r, R_2\restriction_r)$ for every rank r.

Proof
Given two sequences S_1 and S_2 of the correct type with respect to R_1 and R_2 and such that $\mathit{Comp}(S_1, S_2)$, since $\mathit{Comp}(R_1, R_2)$, we have $R_1(S_1) \cap R_2(S_2) = \emptyset$. Let's suppose that (q_1, r_1) is in $R_1\restriction_r(S_1)$, then either $r_1 > r$ and (q_1, r_1) is in $R_1(S_1)$ so that (q_1, r_1) is neither in $R_2(S_2)$ nor in $R_2\restriction_r(S_2)$; or $r_1 \leq r$ and (q_1, r) is in $R_1(S_1)$ so that (q_1, r) is not in $R_2(S_2)$ and (q_1, r_1) is not in $R_2\restriction_r(S_2)$. Similarly we get that whenever (q_2, r_2) is in $R_2\restriction_r(S_2)$ it is not in $R_1\restriction_r(S_1)$. Therefore, we finally have that $R_1\restriction_r(S_1) \cap R_2\restriction_r(S_2) = \emptyset$. Since S_1, S_2 were arbitrary, we get $\mathit{Comp}(R_1\restriction_r, R_2\restriction_r)$. □

The invariant can be now formulated as

$$u \text{ is } q : (x, \rho, \vec{C}) \qquad v \text{ is } (x, \theta) : \vec{R} \to q$$
$$\mathit{Comp}(\theta, \mathit{res}(\rho, u)) \quad \text{and} \quad \mathit{Comp}(\vec{R}, \mathit{res}(\vec{C}, u)) \ .$$

Above, and in the following, the residuals $R(u)$, $\mathit{res}(u, u')$ are calculated with respect to the tree σ_{Adam}.

Examining rules one by one, we will show how Adam can play in order to preserve this invariant, and win.

The case of a variable The play reaches a node v and we have an associated node u satisfying:

$$u \text{ is } q : (x, \rho, \vec{C}) \qquad v \text{ is } (x, \theta) : \vec{R} \to q$$
$$\mathit{Comp}(\theta, \mathit{res}(\rho, u)) \quad \text{and} \quad \mathit{Comp}(\vec{R}, \mathit{res}(\vec{C}, u))$$

Let $\rho(x) = (u_x, N, \rho')$. By the same argument as in the case for the Eve we have $(q, 0) \in R(u_x)\restriction_{\max(u_x,u)}(\vec{S})$ where $\vec{S} = \mathit{res}(\vec{C}, u)$. The invariant tells us that $\mathit{Comp}(\theta(x), R(u_x)\restriction_{\max(u_x,u)})$ and $\mathit{Comp}(\vec{R}, \vec{S})$ hold. In this case by the definition of Comp we have $\theta(x)(\vec{R}) \cap R(u)\restriction_{\max(u_x,u)}(\vec{S}) = \emptyset$. This gives required $(q, 0) \notin \theta(x)(\vec{R})$ showing that Adam wins in v.

The case of a constant The situation is

$$u \text{ is } q : (a, \rho, C_1 \dots C_k) \qquad v \text{ is } (a, \theta) : R_1 \to \dots \to R_k \to q$$
$$\mathit{Comp}(\theta, \mathit{res}(\rho, u)) \quad \text{and} \quad \mathit{Comp}(\vec{R}, \vec{C}) \ .$$

We need to show that node v is loosing for Eve. For this we take a transition $(q_1, \dots, q_k) \in \delta_\alpha(q, a)$ and show that there is $i = 1, \dots, k$ with $(q_i, 0) \notin R_i\lfloor_{rk(q_i)}$. Observe that if there is no such transition then Adam wins. This happens in particular when c is of type o, so also $k = 0$. In this case we cannot have $\emptyset \in \delta(q, a)$, since otherwise u would be loosing for Adam, and the invariant tells that u is a part of his winning strategy.

In the winning strategy of Adam, σ_{Adam}, node u has a successor u' labelled $(q_1, \dots, q_k) : (a, \rho, C_1 \dots C_k)$. Then u' has exactly one successor u'_i labelled $(q_i, u_i) : (N_i, \rho_i, \varepsilon)$, where $C_i = (u_i, N_i, \rho_i)$. We claim that $(q_i, 0) \notin R_i\lfloor_{rk(q_i)}$.

By definition of $R(u_i)$ we have $(q_i, \max(u_i, u'_i)) \in R(u_i)$. So $(q, 0) \in R(u_i)\lfloor_{\max(u_i, u'_i)} = (R(u_i)\lfloor_{\max(u_i,u)})\lfloor_{rk(q_i)}$. Then $\mathit{Comp}(R(u_i)\lfloor_{\max(u_i,u)}, R_i)$ holds by the invariant, and $\mathit{Comp}((R(u_i)\lfloor_{\max(u_i,u)})\lfloor_{rk(q_i)}, R_i\lfloor_{rk(q_i)})$ follows by Lemma 12. So $(q_i, 0) \notin R_i\lfloor_{rk(q_i)}$. This means that Eve cannot conclude by taking the transition $(q, a, q_1, \dots, q_k)$. As the choice of the transition was arbitrary, position v is loosing for Eve.

The case of application The situation is

$$u \text{ is } q : (NK, \rho, \vec{C}) \qquad v \text{ is } (NK, \theta) : \vec{R} \to q$$
$$\mathit{Comp}(\theta, \mathit{res}(\rho, u)) \quad \text{and} \quad \mathit{Comp}(\vec{R}, \mathit{res}(\vec{C}, u)) \ .$$

Then u has the unique successor u' labelled by $q : (N, \rho, (u, K, \rho)\vec{C})$. We need to define the strategy for Adam for every possible choice of P in the application rule applied to v. We have two cases

If $\mathit{Comp}(P, R(u))$ holds then Adam can move to v' labelled $(N, \theta) \geq P \to \vec{R} \to q$, and take u' as the companion node. The invariant will be satisfied as the transition from u to u' has rank 0 so $\mathit{res}(\rho, u) = \mathit{res}(\rho, u')$, $\mathit{res}(\vec{C}, u) = \mathit{res}(\vec{C}, u')$, and $R(u) = \mathit{res}((u, K, \rho), u')$.

If $\mathit{Comp}(P, R(u))$ does not hold then there are $\vec{S}_1, \vec{S}_2$ with $\mathit{Comp}(\vec{S}_1, \vec{S}_2)$ and $R(u)(\vec{S}_1) \cap P(\vec{S}_2) \neq \emptyset$. Let (q', r') be an element from this intersection. From $(q', r') \in R(u)(\vec{S}_1)$ we know that there is a descendant u_1 of u labelled $(q', u) : (K, \rho, \vec{C}_1)$ with $\mathit{res}(\vec{C}_1, u_1) = \vec{S}_1$, and $r' = \max(u, u_1)$. Adam then should take $(K, \theta) \geq P$, followed by $(K, \theta) \geq_{r'} \vec{S}_2 \to q'$, and later by $(K, \theta\lfloor_{r'}) \geq \vec{S}_2 \to q'$. Call this last node v_2. We claim that we can take the unique descendant u_2 of u_1 as the companion of v_2. Indeed u_2 is labelled by $q' : (K, \rho, \vec{C}_1)$. We already know that $\mathit{Comp}(\vec{S}_2, \mathit{res}(\vec{C}_1, u_2))$ holds. Lemma 12 gives $\mathit{Comp}(\theta\lfloor_{r'}, \mathit{res}(\rho, u_2))$, since $\mathit{res}(\rho, u_2) = \mathit{res}(\rho, u)\lfloor_{r'}$.

The case of abstraction and fixpoint In these cases the rules are deterministic, and it is straightforward to check that the invariant is preserved.

Every infinite play is winning As in the case of transferring strategies of Eve, the above construction reflects the ranks, hence if every play in σ_{Adam} is winning for Adam, so is every play in the strategy constructed above.

5 Monotone games

In this section we show that monotone residuals are enough to play in $\mathcal{G}(\mathcal{A}, M)$. By definition (cf. Definition 5) a residual of type $A \to B$ can be any function from $\mathcal{R}_A$ to $\mathcal{R}_B$. Here we show that we can restrict residuals to be monotone functions. The resulting game, that we will call $\mathcal{G}^{mon}(\mathcal{A}, M)$, is in principle more difficult for Eve. Yet, Proposition 16 says that it is in fact equivalent to $\mathcal{G}(\mathcal{A}, M)$.

The set of monotone residuals is defined by:

$$\mathcal{D}_o = \mathcal{P}(Q \times [m]) \qquad \mathcal{D}_{A\to B} = \mathcal{D}_A \xrightarrow{mon} \mathcal{D}_B$$

where $\mathcal{D}_o$ is ordered by inclusion, and $\mathcal{D}_A \xrightarrow{mon} \mathcal{D}_B$ denotes the set of monotone functions from A to B. The set $\mathcal{D}_{A\to B}$ is ordered pointwise. We will use f, g, h to range over elements $\{\mathcal{D}_A\}_{A\in Types}$, υ for a valuation assigning elements of $\{\mathcal{D}_A\}_{A\in Types}$ to variables, and $\vec{g}, \vec{h}$ for sequences of elements of $\{\mathcal{D}_A\}_{A\in Types}$.

Clearly every monotone residual is a residual: $\mathcal{D}_A \subseteq \mathcal{R}_A$ for every type A. We first observe that $\lfloor_r$ operation preserves monotonicity:

Lemma 13 For every monotone residual $R \in \mathcal{D}_A$, and every $r \in [m]$, we have $R\lfloor_r \in \mathcal{D}_A$.

We define an operation mapping a residual to a monotone one.

Definition 14 For type A and every $R \in \mathcal{R}_A$ we define $mon(R) \in \mathcal{D}_A$:

$$\begin{aligned} mon(R) &= R && \text{for } A = o \\ mon(R)(h) &= \bigvee\{mon(R(S)) : mon(S) \leq h\} && \text{for } A = B \to C \text{ and } h \in \mathcal{D}_B \end{aligned}$$

We will need some simple properties of $mon(R)$ operation

Lemma 15 For every type A and residual $R \in \mathcal{R}_A$:

$$mon(R) \in \mathcal{D}_A, \text{ and for every } r \in [m],\ mon(R\lfloor_r) = mon(R)\lfloor_r.$$

Proof
It is clear from the definition that $mon(R)$ is a monotone function. The second statement is proved by the following calculation:

$$\begin{aligned} mon(R\lfloor_r)(\vec{h}) =& \bigcup\{R\lfloor_r(\vec{S}) : mon(\vec{S}) \leq \vec{h}\} \\ =& \bigcup\{(R(\vec{S}))\lfloor_r : mon(\vec{S}) \leq \vec{h}\} \\ =& \big(\bigcup\{R(\vec{S}) : mon(\vec{S}) \leq \vec{h}\}\big)\lfloor_r = mon(R)\lfloor_r(\vec{h}) \end{aligned}$$

The second equality is direct from the definition of the operation $(\cdot)\lfloor_r$. The third from the observation that $(R_1 \cup R_2)\lfloor_r = R_1\lfloor_r \cup R_2\lfloor_r$. □

The game $\mathcal{G}^{mon}(\mathcal{A}, M)$ is defined by the same rules as $\mathcal{G}(\mathcal{A}, M)$ but with the requirement that positions should use only monotone residuals. In particular in the application rule Eve can choose only a monotone residual. In consequence, it is more difficult for Eve to win in $\mathcal{G}^{mon}(\mathcal{A}, M)$ since she has less choice than in $\mathcal{G}(\mathcal{A}, M)$. The next proposition says that despite this the two games are equivalent.

Proposition 16 Suppose $(N, \theta) \geq \vec{R} \to q$ is a winning position for Eve in $\mathcal{G}(\mathcal{A}, M)$. For every $\upsilon \geq mon(\theta)$ and $\vec{f} \geq mon(\vec{R})$ the position $(N, \upsilon) \geq \vec{f} \to q$ is winning for Eve in $\mathcal{G}^{mon}(\mathcal{A}, M)$.

Proof
From a winning strategy for Eve in $\mathcal{G}(\mathcal{A}, M)$ we construct a winning strategy for Eve in $\mathcal{G}^{mon}(\mathcal{A}, M)$. This construction is done rule by rule. The only non-trivial case is that of the application rule. Suppose a position in $\mathcal{G}^{mon}(\mathcal{A}, M)$ is $(NK, \upsilon) \geq \vec{f} \to q$ and we have a position in $\mathcal{G}(\mathcal{A}, M)$ of the form $(NK, \theta) \geq \vec{R} \to q$ with the properties assumed in the statement. Since the later position is winning for Eve, her winning strategy gives her a residual P, and then Adam can decide to move either to $(NK, \theta) \geq P \to \vec{R} \to q$ or to $(K, \theta) \geq P$.

We claim that a good strategy for Eve in the game $\mathcal{G}^{mon}(\mathcal{A}, M)$ is to choose $mon(P)$. Then Adam can choose to move to $(N, \upsilon) \geq mon(P) \to \vec{f} \to q$ or to $(N, \upsilon) \geq mon(P)$. In the first case we can take $(NK, \theta) \geq P \to \vec{R} \to q$ as the associated position and the invariant required in the statement is satisfied.

In the second case, we need to find a position to associate for every position $(K, \upsilon\lfloor_{r_1}) \geq \vec{f_1} \to q_1$ for every $\vec{f_1}$ and $(q_1, r_1) \in mon(P)(\vec{f_1})$. Looking at the definition of $mon(P)$ we have $\vec{S}$ such that $mon(\vec{S}) \leq \vec{f_1}$ and $(q_1, r_1) \in P(\vec{S})$. Since $(K, \theta) \geq P$ is winning for Eve in $\mathcal{G}(\mathcal{A}, M)$ we know that the position $(K, \theta\lfloor_{r_1}) \geq \vec{S} \to q_1$ is winning. Since $\upsilon \geq mon(\theta)$ we have $\upsilon\lfloor_{r_1} \geq mon(\theta\lfloor_{r_1})$ by Lemma 15. So the invariant from the statement is satisfied.

It remains to check that with the so-defined strategy Eve indeed wins in $\mathcal{G}^{mon}(\mathcal{A}, M)$. If a play is infinite then this is indeed the case as the corresponding play in $\mathcal{G}(\mathcal{A}, M)$ is winning and the two plays use the same

moves. It remains to check what happens if the play ends in a variable or constant axiom

For the variable case the play reaches $(x, \upsilon) \geq \vec{f} \to q$ and we know that there is a position $(x, \theta) \geq \vec{R} \to q$ that is winning for Eve, and such that $\vec{f} \geq mon(\vec{R})$ and $\upsilon \geq mon(\theta)$. Since the position is winning for Eve, we have $(q, 0) \in \rho(x)(\vec{R})$. By definition of $mon(\rho(x))$ this gives: $(q, 0) \in mon(\rho(x)(\vec{f})$. Then by monotonicity we get the required $(q, 0) \in \upsilon(x)(\vec{f})$.

For the constant case the play reaches $(a, \upsilon) \geq f_1 \to \cdots \to f_k \to q$ and we know that there is a position $(a, \upsilon) \geq R_1 \to \cdots \to R_k \to q$ that is winning for Eve, and such that $\upsilon \geq mon(\theta)$, and $f_i \geq mon(R_i)$ for $i = 1, \ldots, k$. This gives us a transition $(q, a, q_1, \ldots, q_k) \in \delta_{\mathcal{A}}$ such that $(q_i, 0) \in R_i\restriction_{rk(q_i)}$, for $i = 1, \ldots, k$. But then by Lemma 15 $f_i\restriction_{rk(q_i)} \geq mon(R_i\restriction_{rk(q_i)})$, so we get desired $(q_i, 0) \in f_i\restriction_{rk(q_i)}$. □

Corollary 17 Eve wins from $(M, \emptyset) \geq q$ in $\mathcal{G}(\mathcal{A}, M)$ iff she wins from this position in $\mathcal{G}^{mon}(\mathcal{A}, M)$.

6 Model

The monotone game from the last section can be directly translated into a model. We will use two semantical domains for every type: one with ranks and one without ranks. The results about games make the definition of the model rather direct. The main challenge is to define the interpretation of the fixpoint.

Let us fix a parity automaton $\mathcal{A}$ with Q the set of states, and m the highest rank in its acceptance conditions.

We will work with two semantical domains for every type: domain $\mathcal{D}_A$ as defined in the previous section and another domain:

$$\mathcal{S}_o = \mathcal{P}(Q) \qquad \mathcal{S}_{A \to B} = \mathcal{D}_A \xrightarrow{mon} \mathcal{S}_B$$

where $\mathcal{S}_o$ is ordered by inclusion, and $\mathcal{S}_{A \to B}$ is ordered pointwise. Observe that $\mathcal{S}_{A_1 \to \cdots \to A_n \to o}$ is $\mathcal{D}_{A_1} \to \cdots \to \mathcal{D}_{A_n} \to \mathcal{S}_o$. For every type A, the domain $\mathcal{S}_A$ is a finite complete lattice.

Our goal is Theorem 19 below, stating a direct correspondence between the value of a term M in the model and winning in $\mathcal{G}^{mon}(\mathcal{A}, M)$.

We define the semantics, $[\![M, \upsilon]\!]$ of a term M with respect to a valuation υ assigning elements of $\{\mathcal{D}_A\}_{A \in Types}$ to variables. Semantics $[\![M, \upsilon]\!]$ will be in $\{\mathcal{S}_A\}_{A \in Types}$. At the same time we define auxiliary semantics $\langle\!\langle M, \upsilon \rangle\!\rangle$ that will be in $\{\mathcal{D}_A\}_{A \in Types}$. The two notions are closely related:

$$\begin{aligned} \langle\!\langle M, \upsilon \rangle\!\rangle(\vec{f})(q) &= \{(q, r) : q \in [\![M, \upsilon\restriction_r]\!](\vec{f})\} \\ [\![M, \upsilon]\!](\vec{f}) &= \{q : (q, 0) \in \langle\!\langle M, \upsilon \rangle\!\rangle(\vec{f})\} \ . \end{aligned}$$

The first of these equations should be considered as the definition of $\langle\!\langle M, \upsilon \rangle\!\rangle$, while the second is an immediate consequence of the definition.

It will be useful to define some operations reflecting the above equivalences. For every type A, we define an operation $g \cdot r$ for every $g \in \mathcal{S}_A$ and $r \in [m]$, as well as an operation $f^{(0)}$ for every $f \in \mathcal{D}_A$:

$$\begin{aligned}(g \cdot r)(\vec{h}) =& \{(q, r) : q \in g(\vec{h})\} \\ f^{(0)}(\vec{h}) =& \{q : (q, 0) \in f(\vec{h})\} \ .\end{aligned}$$

So $g \cdot r$ is an element of $\mathcal{D}_A$, and $f^{(0)}$ is an element of $\mathcal{S}_A$. In particular we have:

$$[\![M, \upsilon]\!] = (\langle\!\langle M, \upsilon \rangle\!\rangle)^{(0)} \quad \text{and} \quad \langle\!\langle M, \upsilon \rangle\!\rangle = \bigvee_{r=0}^{m} [\![M, \upsilon\lfloor_r]\!] \cdot r$$

With these definitions we are ready to write the semantical clauses:

$$\begin{aligned}[\![x, \upsilon]\!]\vec{f} =& \{q : (q, 0) \in \upsilon(x)(\vec{f})\} \\ [\![a, \upsilon]\!] f_1 \dots f_k =& \{q : \exists_{(q_1,\dots,q_k)\in\delta_{\mathcal{A}}(q,a)}.\ \forall_{i=1,\dots,k}.\ (q_i, 0) \in f_i\lfloor_{rk(q_i)}\} \\ [\![\lambda x.M, \upsilon]\!] f =& [\![M, \upsilon[f/x]]\!] \\ [\![MN, \upsilon]\!] =& [\![M, \upsilon]\!]\langle\!\langle N, \upsilon \rangle\!\rangle \\ [\![Y, \upsilon]\!] f =& \mathrm{fix}(f, 0) \qquad \text{where for } l = 0, \dots, m \text{ we have} \\ \mathrm{fix}(f, l) =& \sigma g_l \dots \mu g_1 . \nu g_0 .\ (f\lfloor_l)^{(0)}(\bigvee_{r=0}^{l} g_r \cdot r \vee \bigvee_{r=l+1}^{m} \mathrm{fix}(f, r) \cdot r)\end{aligned}$$

Observe that the clause for the variable can be written in a shorter way as $[\![x, \upsilon]\!] = (\upsilon(x))^{(0)}$. The internal formula in the definition of fix also resembles the relations between $[\![M, \upsilon]\!]$ and $\langle\!\langle M, \upsilon \rangle\!\rangle$.

Proposition 18 For every term M and valuation υ both $[\![M, \upsilon]\!]$ and $\langle\!\langle M, \upsilon \rangle\!\rangle$ are monotone functions. If $M =_{\beta\delta} N$ then $[\![M, \upsilon]\!] = [\![N, \upsilon]\!]$ and $\langle\!\langle M, \upsilon \rangle\!\rangle = \langle\!\langle N, \upsilon \rangle\!\rangle$.

Proof
The first statement follows easily by induction on the structure of the term. In particular the semantics of Y is a monotone function since it is defined as a composition of monotone operations.

In order to prove that the model is sound with respect to β and δ reductions, we will need first to spell out some properties of the semantics. The first, explains the role of $\lfloor_r$ operation.

$$\langle\!\langle K, \upsilon\lfloor_r \rangle\!\rangle = \langle\!\langle K, \upsilon \rangle\!\rangle\lfloor_r. \tag{1}$$

This property follows from a short calculation, where the equation in the middle uses the Lemma 7 saying that $(f\lfloor_r)\lfloor_i = f\lfloor_{\max(r,i)}$:

$$\langle\!\langle K, v\lfloor_r \rangle\!\rangle = \bigvee_{i=1}^{m} [\![K, (v\lfloor_r)\lfloor_i]\!] \cdot i = \\ = \bigvee_{i=0}^{r} [\![K, v\lfloor_r]\!] \cdot_i \vee \bigvee_{i=r+1}^{m} [\![K, v\lfloor_i]\!] \cdot i = \langle\!\langle K, v\rangle\!\rangle\lfloor_r$$

The next observation is that substitution is well behaved:

$$[\![M[K/x], v]\!] = [\![M, v[\langle\!\langle K, v\rangle\!\rangle/x]]\!] \quad \text{and} \quad \langle\!\langle M[K/x], v\rangle\!\rangle = \langle\!\langle M, v[\langle\!\langle K, v\rangle\!\rangle/x]\rangle\!\rangle \tag{2}$$

The proof of these two statements is done by mutual induction on the structure of M. The statetement for $[\![M[K/x], v]\!]$ follows directly from induction assumption, while the statement for $\langle\!\langle M[K/x], v\rangle\!\rangle$ follows from the statement for $[\![M[K/x], v]\!]$ as the following calculation shows

$$\begin{aligned} \langle\!\langle M[K/x], v\rangle\!\rangle &= \bigvee_{r=0}^{m} [\![M[K/x], v\lfloor_r]\!] \cdot r \\ &= \bigvee_{r=0}^{m} [\![M, v\lfloor_r[[\![K, v\lfloor_r]\!]/x]]\!] \cdot r \\ &= \bigvee_{r=0}^{m} [\![M, v\lfloor_r[[\![K, v]\!]\lfloor_r/x]]\!] \cdot r \\ &= \bigvee_{r=0}^{m} [\![M, (v[[\![K, v]\!]/x])\lfloor_r]\!] \cdot r = \langle\!\langle M, (v[[\![K, v]\!]/x])\lfloor_r\rangle\!\rangle \end{aligned}$$

The second equality above follows from the hypothesis; the third from equation (1) we have just proved; the others follow from definitions.

Equations (2) allow us to prove the soundness of the model with respect to β-reduction. Consider $[\![(\lambda x.M)N, v]\!]$. By the semantics this it is equal to $[\![M, v[\langle\!\langle N, v\rangle\!\rangle/x]]\!]$ and then equation (2) tells us that it is the same as $[\![M[N/x], v]\!]$.

In order to prove soundness with respect to δ-reduction we will need one more observation:

$$\mathrm{fix}(f\lfloor_k, 0) = \mathrm{fix}(f, k) \tag{3}$$

For this we show that $\mathrm{fix}(f\lfloor_r, r-1) = \mathrm{fix}(f, r)$ using the following calcula-

tion:

$$\begin{aligned}
\mathrm{fix}(f,r) &= \sigma g_r \ldots \mu g_1.\nu g_0.\ (f\lfloor_r)^{(0)}\big(\bigvee_{i=1}^{r} g_i\cdot i \vee \bigvee_{i=r+1}^{m} \mathrm{fix}(f,i)\cdot i\big)\\
&= \sigma' g_{r-1}\cdots\mu g_1.\nu g_0.\ (f\lfloor_r)^{(0)}\big(\bigvee_{i=1}^{r-1} g_i\cdot i \vee \bigvee_{i=r}^{m} \mathrm{fix}(f,i)\cdot i\big)\\
&= \sigma' g_{r-1}\cdots\mu g_1.\nu g_0.\ ((f\lfloor_r)\lfloor_{r-1})^{(0)}\big(\bigvee_{i=1}^{r-1} g_i\cdot i \vee \bigvee_{i=r}^{m} \mathrm{fix}(f,i)\cdot i\big)\\
&= \mathrm{fix}(f\lfloor_r, r-1)
\end{aligned}$$

Now using this observation we have for $k>0$: $\mathrm{fix}(f\lfloor_k,0) = \mathrm{fix}((f\lfloor_k)\lfloor_1,0) = \mathrm{fix}(f\lfloor_k,1)$. Applying this argument $k-1$ times we get $\mathrm{fix}(f\lfloor_k,0) = \mathrm{fix}(f\lfloor_k,k-1)$, and then we can once again use the observed equality to obtain (3).

Equation (3) allows us to prove soundness with respect to δ reduction as follows:

$$\begin{aligned}
[\![M(YM),\upsilon]\!] &= [\![M,\upsilon]\!](\langle\!\langle YM,\upsilon\rangle\!\rangle)\\
&= [\![M,\upsilon]\!]\left(\bigvee_{i=0}^{m} [\![YM,\upsilon\lfloor_i]\!]\cdot i\right)\\
&= [\![M,\upsilon]\!]\left(\bigvee_{i=0}^{m} \mathrm{fix}(\langle\!\langle M,\upsilon\lfloor_i\rangle\!\rangle,0)\cdot i\right)\\
&= [\![M,\upsilon]\!]\left(\bigvee_{i=0}^{m} \mathrm{fix}(\langle\!\langle M,\upsilon\rangle\!\rangle\lfloor_i,0)\cdot i\right) && \text{by equation (1)}\\
&= [\![M,\upsilon]\!]\left(\bigvee_{i=0}^{m} \mathrm{fix}(\langle\!\langle M,\upsilon\rangle\!\rangle,i)\cdot i\right) && \text{by equation (3)}\\
&= (\langle\!\langle M,\upsilon\rangle\!\rangle)^{(0)}\left(\bigvee_{i=0}^{m} \mathrm{fix}(\langle\!\langle M,\upsilon\rangle\!\rangle,i)\cdot i\right) && \text{by definition}\\
&= \mathrm{fix}(\langle\!\langle M,\upsilon\rangle\!\rangle,0) = [\![Y,\upsilon]\!](\langle\!\langle M,\upsilon\rangle\!\rangle) = [\![YM,\upsilon]\!]
\end{aligned}$$

□

Theorem 19 *For a given automaton $\mathcal{A}$ the model constructed above is such that for every term M, state q, valuation υ, and a sequence of residuals $\vec{f}$:*

$$q \in [\![M,\upsilon]\!](\vec{f}) \text{ iff } (M,\upsilon)\geq\vec{f}\to q \text{ is winning for Eve in } \mathcal{G}^{mon}(\mathcal{A},M).$$

Before proving the theorem let us see how it implies our main result, i.e., Theorem 1. Let M be a closed term of type o. Without a loss of generality we can assume that Y appears only in applicative contexts in M

as discussed in the remark on Page 3. If it is not the case then we replace Y with $\lambda x.Yx$. Clearly this operation does not change the semantic value of the term. Now suppose $q \in [\![M, \emptyset]\!]$. The above theorem says that this happens if and only if $(M, \emptyset) \geq q$ is winning for Eve in $\mathcal{G}^{mon}(\mathcal{A}, M)$. By Corollary 17 the later is equivalent to $(M, \emptyset) \geq q$ being winning for Eve in $\mathcal{G}(\mathcal{A}, M)$. Then Theorem 9 gives equivalence with winning from $q : (M, \emptyset, \varepsilon)$ in the game $\mathcal{K}(\mathcal{A}, M)$. Finally, Proposition 4, says that this is the same as $BT(M)$ being accepted by the automaton $\mathcal{A}$ from the state q.

The rest of this section is devoted to the proof of Theorem 19. The proof is by induction on the structure of M. The cases of a variable, a constant, and λ-abstraction are direct from the definitions. We first consider the application case leaving, the most difficult, fixpoint case for the end.

To prove the left to right implication of the application case take $q \in [\![MN, \upsilon]\!](\vec{f}) = [\![M, \upsilon]\!]\langle\!\langle N, \upsilon \rangle\!\rangle \vec{f}$. Then the position $(M, \upsilon) \geq \langle\!\langle N, \upsilon \rangle\!\rangle \to \vec{f} \to q$ is winning by the induction hypotheis. We need to show that $(N, \upsilon) \geq \langle\!\langle N, \upsilon \rangle\!\rangle$ is winning. For this we take arbitrary $\vec{h}_1$ and $(q_1, r_1) \in \langle\!\langle N, \upsilon \rangle\!\rangle(\vec{h}_1)$. By definition $q_1 \in [\![N, \upsilon\restriction_{r_1}]\!](h_1)$. So $(N, \upsilon\restriction_{r_1}) \geq \vec{h}_1 \to q_1$ is winning by the induction hypothesis.

For the right to left implication of the application case take a winning position $(MN, \upsilon) \geq \vec{f} \to q$. The winning strategy in the game gives us the following situation for some monotone residual g:

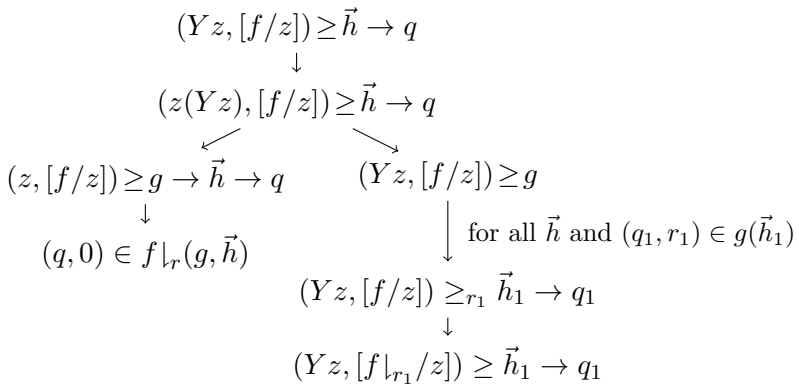
$$\begin{array}{ccc} & (MN, \upsilon) \geq \vec{f} \to q & \\ \swarrow & & \searrow \text{for all } \vec{h}_1 \text{ and } (q_1, r_1) \in g(\vec{h}_1) \\ (M, \upsilon) \geq g \to \vec{f} \to q & & (N, \upsilon\restriction_{r_1}) \geq \vec{h}_1 \to q_1 \end{array}$$

The induction hypothesis tells us $q \in [\![M, \upsilon]\!](g, \vec{f})$. We will show $g \leq \langle\!\langle N, \upsilon \rangle\!\rangle$, that by monotonicity will imply: $q \in [\![M, \upsilon]\!](\langle\!\langle N, \upsilon \rangle\!\rangle, \vec{f}) = [\![MN, \upsilon]\!](\vec{f})$.

To show $g \leq \langle\!\langle N, \upsilon \rangle\!\rangle$ we take arbitrary $\vec{h}_1$ and $(q_1, r_1) \in g(\vec{h}_1)$. By the assumption $(N, \upsilon\restriction_{r_1})) \geq \vec{h}_1 \to q_1$ is winning. So $(q_1, r_1) \in \langle\!\langle N, \upsilon \rangle\!\rangle(\vec{h}_1)$ by the induction hypothesis.

For the fixpoint case we will need some preparations. Let $A = \vec{B} \to o$ be some type. We want to show the theorem for the fixpoint operator $Y : (A \to A) \to A$.

Consider game $\mathcal{G}^{mon}(\mathcal{A}, M)$ from a position $(Yz, [f/z]) \geq \vec{h} \to q$.



In the above game first Eve chooses a monotone residual g, and then Adam chooses a sequence of monotone residuals $\vec{h}_1$ and $(q_1, r_1) \in g(h_1)$. The left branch ends with a test giving a restriction on what residual g Eve can choose. From the right branch the game continues, so in the course of the game f will accumulate $\lfloor_r$ operations resulting in: $(\dots(f\lfloor_{r_1})\dots\lfloor_{r_k}) = f\lfloor_{\max(r_1,\dots,r_k)}$.

In order to make the notation lighter we introduce another game that behaves exactly in the same way as the one above. We call it G_A^{Yf}:

- Eve's positions $(q, l, \vec{h})$ with $\vec{h} \in \mathcal{D}_{\vec{B}}$;
- Adam's positions $[g, l]$ with $g \in \mathcal{D}_A$, and $l \in [m]$.

The transitions of G_A^{Yf} are

- $(q, l, \vec{h}) \to [g, l]$ for $g \in D_A$ such that $(q, 0) \in f\lfloor_l(g, \vec{h})$, this transition has rank 0;
- $[g, l] \to (q, \max(l, r), \vec{h})$ for $(q, r) \in g(\vec{h})$, this transition has rank r.

Eve chooses in positions of the form $(q, l, \vec{h})$ and Adam in positions of the form $[g, l]$.

The next lemma describes the relation between two games.

Lemma 20 Eve wins form a position $(q, l, \vec{h})$ in G_A^{Yf} iff she wins from the position $(Yz, [f\lfloor_l/z]) \geq \vec{h} \to q$ in $\mathcal{G}^{mon}(\mathcal{A}, M)$.

Thanks to the lemma, we can focus on the relation between the semantics and the game G_A^{Yf}.

We can consider the game G_A^{Yf} as a transition system with positions as states and moves as transitions. Every transition has a label that is its rank. We will use a notation borrowed from modal logic. For a set of positions V we write $\langle 0\rangle V$ for the set of positions having a transition of rank 0 leading to V. Dually, $[i]V$ is the set of positions from which every transition of

rank i leads to a position from V. With this notation the set of winning positions for Eve in the game G_A^{Yf} in given by the following fixpoint formula (we suppose that m is odd to simplify the notation).

$$\theta_{win} = \mu X_m.\nu X_{m-1} \ldots \mu X_1.\nu X_0.\ \langle 0 \rangle \left(\bigwedge_{i=0}^{m} [i] X_i \right) \tag{4}$$

In the argument below we will use step functions. For $\vec{h} \in \mathcal{D}_{\vec{B}}$ and $q \in \mathcal{S}_0$, we write $(\vec{h} \mapsto b)$ for the function in $D_{\vec{B} \to o}$ defined by:

$$(\vec{h} \mapsto q)(\vec{x}) = \begin{cases} q \text{ when } \vec{h} \leq \vec{x} \\ \bot \text{ otherwise} \end{cases}$$

These preparations show that in order to prove the theorem for the case of the fixpoint operator it is enough to show:

Lemma 21 For every l: $\mathrm{fix}(f, l) \geq \vec{h} \mapsto q$ iff $(q, l, \vec{h}) \in \theta_{win}$.

The proof of this lemma is by induction on l, but first we need some preparations.

Definition 22 For every set of positions V of Eve in the game G_A^{Yf} and every $r \in [m]$ we define a function:

$$[V, r] = \bigvee \{ \vec{h} \mapsto \{q\} : (q, r, \vec{h}) \in V \} \ .$$

The next lemma shows a relation between internal part of the formula (4), and the internal part of the formula defining the fixpoint.

Lemma 23 Let $V_0, \ldots, V_m$ be sets of positions of Eve in the game G_A^{Yf}. For every state q, every $l \in [m]$, and every vector of monotone residuals $\vec{h}$ of appropriate types:

$$(f\lfloor_l)^{(0)}\big(\bigvee_{r=0}^{l} [V_r, l] \cdot r \vee \bigvee_{r=l+1}^{m} [V_r, r] \cdot r \big) \geq \vec{h} \mapsto \{q\}$$

iff

$$(q, l, \vec{h}) \in \langle 0 \rangle \big(\bigwedge_{r=0}^{m} [r] V_r \big)$$

Proof
Let $g = \bigvee_{r=0}^{l} [V_r, l] \cdot r \vee \bigvee_{r=l+1}^{m} [V_r, r] \cdot r$. We first consider left to right direction.

Suppose $(f\lfloor_l)^{(0)}(g) \geq \vec{h} \to q$. This means that $(q, 0) \in f\lfloor_l(g, \vec{h})$. So from $(q, l, \vec{h})$ Eve can take a transition to $[g, l]$. We need to show that $[g, l] \in \bigwedge_{r=0}^{m} [r] V_r$.

Suppose Adam takes a transition of rank r_1 to $(q_1, \max(l, r_1), \vec{h}_1)$. From the definition of the game we get $(q_1, r_1) \in g(\vec{h}_1)$. Let us look at the definition of g. If $r_1 \leq l$ then $(q_1, r_1) \in [V_{r_1}, l] \cdot r_1$, so $(q_1, l, \vec{h}_1) \in V_{r_1}$. If $r_1 > l$ then $(q_1, r_1) \in [V_{r_1}, r_1] \cdot r_1$ so $(q_1, r_1, \vec{h}_1) \in V_{r_1}$. In both cases we end up in V_{r_1} as required.

For the other direction suppose $(q, l, \vec{h}) \in \langle 0 \rangle \big(\bigwedge_{r=0}^{m} [r] V_r \big)$. Then there is $[g_1, l_1] \in \bigwedge_{r=0}^{m} [r] V_r$ with $(q, 0) \in f\lfloor_l(g_1, \vec{h})$. It remains to show $g_1 \leq g$.

Take $\vec{h}_1$ and $(q_1, r_1) \in g_1(\vec{h}_1)$. By the definition of the game we have $(q, \max(r_1, l), \vec{h}_1) \in V_{r_1}$. If $r_1 \leq l$ then $[V_{r_1}, l] \geq \vec{h} \mapsto \{q\}$, giving $(q, r_1) \in ([V_{r_1}, l] \cdot r_1)(\vec{h}_1)$, and in consequence $(q, r_1) \in g(\vec{h}_1)$. Similarly, if $r_1 > l$ then $[V_{r_1}, r_1] \geq \vec{h} \mapsto \{q\}$, giving $(q, r_1) \in ([V_{r_1}, r_1] \cdot r_1)(\vec{h}_1)$, and in consequence $(q, r_1) \in g(\vec{h}_1)$. This shows that $g_1 \leq g$.

□

By induction we can extend the correspondence from the previous lemma to formulas with fixpoints.

Lemma 24 For every k and l verifying $-1 \leq k \leq l \leq m$, and for all sets $V_{k+1}, \ldots, V_m$ of positions of Eve in G_A^{Yf}:

$$\sigma g_k \ldots \mu g_1 \nu g_0.\ (f\lfloor_l)^{(0)} \big(\bigvee_{r=0}^{k} g_r \cdot r \cup \bigvee_{r=k+1}^{l} [V_r, l] \cdot r \vee \bigvee_{r=l+1}^{m} [V_r, r] \cdot r \big) \geq \vec{h} \mapsto \{q\}$$

iff

$$(q, l, \vec{h}) \in \sigma X_k \ldots \mu X_1 \nu X_0.\ \langle 0 \rangle \big(\bigwedge_{r=0}^{k} [r] X_r \wedge \bigwedge_{r=k+1}^{m} [r] V_r \big)$$

Proof
We fix $l \in [m]$ and we prove the lemma by induction on k. The proof for $k = -1$ is the statement of the previous lemma. The induction step uses the unrolling of the greatest/least fixpoint definitions. □

Taking $k = l$ in the lemma above gives Lemma 21 and finishes the proof of the theorem.

7 Related work

The objective of this paper is to present a complete proof of the model construction for Ω-blind regular properties. Technically, it is just a simplification of our construction from [SW15a] where we treat all regular properties. Nevertheless, we hope that this exercise is useful as most of the literature on the subject concentrates on Ω-blind properties.

The present complete exposition allows us also to put this construction in the context of related research tracing the origins of each of its ingredients.

The first step is a very influential paper of Kobayashi and Ong [KO09], and in particular its type system for calculating maximal ranks that appear on paths of Böhm trees of simply typed λ-terms without recursion. This idea influenced the notion of residual essential in our Krivine machine approach to the higher-order model checking problem [SW11]. All technical development in Sections 3 and 4 comes from [SW11].

In order to facilitate comparison with another very interesting work, namely that of Tsukada and Ong [TO14], we have presented the game $\mathcal{G}(\mathcal{A}, M)$ from [SW11] in a form of a type system. Tsukada and Ong construct an infinite model and use it to show completeness of a type system for higher-order model checking. Their type system is essentially the same as the one defining our game in Section 4. The only difference is in lifting operations: Tsukada and Ong system uses $r\backslash$ while we use $\lfloor_r$. For the fixpoints, the Tsukada and Ong type system does the same thing as our game: it delegates them to an external solving procedure.

To finish the comparison with the Tsukada and Ong type system, let us look closer at the $r\backslash$ operation from [TO14] and compare it to $\lfloor_r$ from [SW11]. The operation $r\backslash$ is defined in a more abstract way but for automata with parity conditions it has a presentation that makes a direct comparison possible. Tsukada and Ong consider the following order on ranks:

$$2n \preccurlyeq 2n-2 \preccurlyeq \ldots \preccurlyeq 2 \preccurlyeq 0 \preccurlyeq 1 \preccurlyeq \ldots \preccurlyeq 2n-1 \preccurlyeq 2n+1$$

Intuitively $i \preccurlyeq j$ means than from the point of view of the parity condition it is better to see i than j. The operation $r\backslash$ is defined so that $r\backslash e$ is the biggest r' in the $\preccurlyeq$ ordering such that $\max(r, r') \preccurlyeq e$; where max is taken in the natural ordering on numbers. In our constructions we work with sets of ranks, while Tsukada and Ong consider only $\preccurlyeq$-downward closed ones: $[e] = \{r : r \preccurlyeq e\}$. The operations $\lfloor_r$ and $r\backslash$ are the same on $\preccurlyeq$-downward closed sets since:

$$[e]\lfloor_r = [r\backslash e]$$

For algorithmic reasons it may be interesting to work with $\preccurlyeq$-downward closed sets. Our model construction preserves downward closures, so we could have imposed $\preccurlyeq$-downward closure as an additional condition on residuals; in this case the two type systems would become identical. We have chosen not to do this since adding $\preccurlyeq$-order does not seem to simplify our constructions.

Recently Grellois and Mellies [GM15a] have given another reduction of the model-checking problem for Ω-blind automata to finite parity games.

Section 5 presents a reduction from arbitrary games to games over monotone residuals. The restriction to monotone residuals is not needed when the fixpoints are treated via games, so it does not appear in other works, but it is essential if we want to give a semantics to the fixpoint operators.

Section 6 gives the model construction that directly reflects games over monotone residuals. This is not to say that the model construction did not require new insights. The most important comes from the work of Grellois and Mellies. In [GM15c] they have given a categorical account of the behaviour of ranks in a model. They derive an infinite model via elegant general constructions. Later Mellies [Mel14] clearly showed the value of using the morphism composition similar to that in Kleisli categories. This composition is used in our model.

8 Conclusions

The model construction presented here is largely based on the construction from [SW11]. The main technical tool is the finite game $\mathcal{G}(M, \mathcal{A})$. To obtain a model it suffices to show that Eve can play with monotone residuals, and to understand the nature of composition as well as the fixpoint operator with respect to the game.

The general case, for parity automata that are not necessary Ω-blind, is more complicated. All the games should be modified, and in consequence the match between games and the monotone model is lost. The solution we propose in [SW15a] is to impose one more restriction on residuals that we call stratification. One can show that in modified games Eve can play with stratified monotone residuals, and that all operations needed to construct a model preserve the stratification property. This unfortunately adds additional level of difficulty to the whole construction.

References

[GM15a] Charles Grellois and Paul-André Melliès. Finitary semantics of linear logic and higher-order model-checking. *CoRR*, abs/1502.05147, 2015. To appear at CSL'15.

[GM15b] Charles Grellois and Paul-André Melliès. An infinitary model of linear logic. In *FOSSACS 15*, volume 9034 of *LNCS*, pages 41–55, 2015.

[GM15c] Charles Grellois and Paul-André Melliès. Tensorial logic with colours and higher-order model checking. *CoRR*, abs/1501.04789, 2015. To appear at MFCS'15.

[Had13] A. Haddad. Model checking and functional program transformations. In *FSTTCS*, volume 24 of *LIPIcs*, pages 115–126, 2013.

[KO09] N. Kobayashi and L. Ong. A type system equivalent to modal mu-calculus model checking of recursion schemes. In *LICS'09*, pages 179–188, 2009.

[Kri07] Jean-Louis Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3):199–207, 2007.

[Mel14] Paul-André Melliès. Linear logic and higher order model-checking. Talk at IHP Wokshop, June 2014.

[Ong06] C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS'06*, pages 81–90, 2006.

[SW11] Sylvain Salvati and Igor Walukiewicz. Krivine machines and higher-order schemes. In *ICALP'11*, volume 6756 of *LNCS*, pages 162–173, 2011.

[SW13] Sylvain Salvati and Igor Walukiewicz. Using models to model-check recursive schemes. In *TLCA*, volume 7941 of *LNCS*, pages 189–204, 2013.

[SW14] Sylvain Salvati and Igor Walukiewicz. Krivine machines and higher-order schemes. *Inf. Comput.*, 239:340–355, 2014.

[SW15a] Sylvain Salvati and Igor Walukiewicz. A model for behavioural properties of higher-order programs. HAL-01145494, to appear at CSL'15, 2015.

[SW15b] Sylvain Salvati and Igor Walukiewicz. Typing weak MSOL properties. In *FOSSACS 15*, volume 9034, pages 343–357, 2015.

[TO14] Takeshi Tsukada and C.-H. Luke Ong. Compositional higher-order model checking via ω-regular games over Böhm trees. In *LICS-CSL*, pages 78:1–78:10, 2014.